Vahvistusoppiminen käsittelyroboteille

Turun yliopisto
Tietotekniikan laitos
LuK-tutkielma
Tietojenkäsittelytiede
Tammikuu 2025
Uuno Tapper

Ohjaaja:
Mika Murtojärvi

TURUN YLIOPISTO
Tietotekniikan laitos

Uuno Tapper: Vahvistusoppiminen käsittelyroboteille

LuK-tutkielma, 28 s.
Tietojenkäsittelytiede
Tammikuu 2025

Vahvistusoppiminen jakautuu mallivapaaseen ja -pohjaiseen vahvistusoppimiseen. Mallivapaat menetelmät pyrkivät parantamaan arvofunktiota ja toimintatapaa. Mallipohjaiset menetelmät luovat tai hyödyntävät malleja ympäristöstä osana päätös- ja opetusprosessia.
Syvävahvistusoppiminen laajentaa vahvistusoppimisen käyttökohteiden mahdollista monimutkaisuutta. Syväoppimista hyödynnetään laajasti robotiikassa datan korkeaulotteisuuden takia. Moni robotti hyödyntää monimutkaisia sensoreita erilaisin konenäön käyttötarkoituksiin.
Tutkielmassa tarkastellaan vahvistusoppimisen teoriaa ja sen hyödynnystä robotiikassa. Tähän hyödynnetään useasti simuloituja robotteja ja ympäristöjä, sillä se mahdollistaa nopean ja turvallisen datan keräämisen. Tutkielma tarkastelee vahvistusoppimisen sovellusalueita robotiikassa käsittelyn kautta. Kirjallisuus nostaa esille vahvistusoppimisen tarpeen monimutkaisten rutiinien oppimiseen, kuten myös robottien sopeuttamisen ympäristön muutoksiin, kun perinteiset ohjelmointimenetelmät eivät riitä.

Asiasanat: Vahvistusoppiminen; Robotiikka; Mallivapaa; Mallipohjainen; Syvävahvistusoppiminen; Simulaatio; Koneoppiminen

Sisällys

1 Johdanto **1**

2 Vahvistusoppiminen **4**

2.1 Koneoppiminen robotiikassa 4

2.2 Vahvistusoppiminen ja Markovin päätösprosessi 6

2.2.1 Mallivapaa vahvistusoppiminen 9

2.2.2 Mallipohjainen vahvistusoppiminen 11

2.2.3 Syvävahvistusoppiminen 13

3 Simulaatio **16**

3.1 Simulaation toteutus 16

3.2 Simulaatioympäristöt 17

4 Vahvistusoppiminen, simulaatio ja robotiikka **19**

4.1 Haasteet 19

4.2 Yleistäminen 20

4.3 Käsittelyrobotiikka 21

4.3.1 Monimutkainen esineiden käsittely 23

4.3.2 Liikkuvat ja yhteistyötä tekevät robotit 24

5 Yhteenveto **26**

Lähdeluettelo **29**

1 Johdanto

Robotit ovat vaihtelevia muodoiltaan ja toiminnoiltaan. Perinteisesti niiden rutiinit ohjelmoidaan liike liikkeeltä. Tämä käy nopeasti työlääksi, koska tarpeet erilaisiin tilanteisiin voivat olla lähes rajattomat. Siksi tarvitaan automaatiojärjestelmiä, jotka kykenevät luomaan tai muokkaamaan rutiineja.

Roboteille on kolme pääasiallista lähestymistapaa rakentaa rutiineja: suora ohjelmointi, imitaatio-oppiminen ja vahvistusoppiminen. Suoran ohjelmoinnin avulla valmistettu järjestelmä toimii tehokkaasti, jos ympäristö on hallittu.[1] Imitaatio-oppimisessa robottia hallitaan joko suoraan tai sille syötetään ohjausmateriaali.[2] Vahvistusoppiminen mahdollistaa robotin mallintaa käytöstä, joka olisi haastavaa ohjelmoida. Vahvistusoppimisen avulla roboteille voidaan mallintaa optimaalista käytöstä, vaikka tätä ei tiedettäisi etukäteen.[3] Robotille etsittävää käytöstä kutsutaan toimintatavaksi (engl. policy). [4]

Vahvistusoppimisella voidaan opettaa roboteille rutiineja itsenäisesti testaamalla. Tätä prosessia voidaan toteuttaa todellisuudessa tai simuloidussa ympäristössä.[4] Näin oppimismallit tarjoavat robotiikassa enemmän joustavuutta järjestelmille kuin perinteiset esiohjelmoidut järjestelmät.[2] Tavoitteena on löytää soveltuva toimintatapa, mitä seurata. Soveltuvien syötteiden räätälöiminen on keskeistä oppimisprosessin laadulle. [3] Näitä syötteitä voivat olla esimerkiksi robotin osien sijainnit ja kamerakuva. Oppimisprosessi säätää toimintatavan painotuksia eri tiloissa (engl. state), mikä muokkaa robotin käyttäytymistä.

Vahvistusoppiminen tarvitsee useita koulutusjaksoja (engl. iteration), mikä vie paljon aikaa ja voi vaurioittaa robottia. Lisäksi kouluttaminen todellisessa ympäristössä saattaa olla vaarallista ihmisille. [4] Tämän takia on hyvin hyödyllistä, että robotti voidaan simuloida erilaisissa ympäristöissä, jotka tuottavat erilaisia rasitteita ja häiriöitä ilman fyysistä vauriota.[5] Simulaation avulla opetus voidaan toteuttaa nopeammin ja useampaan tilanteeseen. Koulutusjaksot voidaan toteuttaa instansseissa, joita pyöritetään simulaatiossa, joka jäljittelee fyysistä ympäristöä.[3] Lisäksi on mahdollista opettaa järjestelmää erilaisilla metatiedoilla (engl. meta-learning), jolloin opetetaan, mikä on tärkeää kyseisessä tehtävässä. [2]

Tämä tutkielma on kirjallisuuskatsaus, jonka tarkoituksena on perehtyä vahvistusoppimiseen robotiikan kontekstissa.

Tutkimuskysymykset, joihin tutkielma pyrkii vastaamaan, ovat:

1. Mitä on vahvistusoppiminen?

2. Miten simulaatiota hyödynnetään tässä prosessissa?

3. Mitä hyötyä vahvistusoppimisesta on robotiikassa?

Tiedonhakuun on käytetty pääasiallisesti Turun yliopiston Volter-tietokantaa ja Google Scholar -palvelua. Tutkimuskysymysten perusteella hakujen muodostukseen nousi keskeiseksi etsiä katsaustutkimuksia aihealueista. Keskeisin hakulauseke oli muotoa ("Reinforcement learning" AND "survey") AND (Robotics OR Simulation). Toisen kysymyksen hakulause oli ("Simulation" OR "Simulator") AND ("Reinforcement learning" OR "Robotics") AND "survey". Kolmannen kysymyksen perusteella rakennettu hakulause oli "Robotics" AND ("Reinforcement learning" OR "Deep learning") AND "Manipulation". Näiden lisäksi vahvistusoppimisesta tarkempien lähteiden haussa käytettiin seuraavia hakulauseita: "Reinforcement learning" AND "Model-free learning" AND (NOT "Model-based learning"), "Reinforcement learning" AND "Model-based learning" AND (NOT "Model-free learning") ja "Rein-

forcement learning" AND "taxonomy". Lähteiden karsinta toteutui rajaamalla hakutuloksia neljään viime vuoteen, tai katsausartikkelien tapauksessa eniten viitattuihin tutkielmiin.

Tutkielman rakenne on seuraavanlainen; toisessa luvussa käsitellään vahvistusoppimista. Kolmannessa luvussa käsitellään simulaatiota lyhyesti. Neljännessä luvussa käsitellään robotiikan ja vahvistusoppimisen yhdistämistä. Viides luku on yhteenveto.

2 Vahvistusoppiminen

2.1 Koneoppiminen robotiikassa

Koneoppiminen on hyödyllinen työkalu välttämään jokaisen tilanteen esiohjelmointia. Järjestelmän rutiinit voidaan sen sijaan opettaa datalla. Yleisesti koneoppiminen jakautuu kolmeen kategoriaan: Ohjattu oppiminen (engl. supervised learning), ohjaamaton oppiminen (engl. unsupervised learning) ja vahvistusoppiminen.[2] Keskeisimmät menetelmät, joita robotiikassa hyödynnetään, ovat jäljittelyoppiminen (engl. imitation learning), kontekstuaalinen rosvo (engl. contextual bandit) ja Markovin päätösprosessiin perustuva vahvistusoppiminen. [1] [6]

Ohjattu oppiminen perustuu hyvin merkittyyn dataan, josta löytyy alku- ja lopputulos. Menetelmä oppii yhdistämään uudet syötteet johonkin datasetin mahdollisista lopputuloksista. Tämä tapahtuu algoritmeilla ja heuristiikalla. Käytännössä menetelmän algoritmeilla on hankaluuksia robotiikassa, koska menetelmä olettaa, että aloitustilanne ei muutu. Jäljittelyoppiminen useasti hyödyntää ohjatun oppimisen menetelmiä. Vaihtoehtoisesti ohjaamattoman oppimisen menetelmät toimivat datan merkkaamattoman datan kanssa. Useasti tämä tarkoittaa, että prosessi on varsin hidas ja epätarkka. Moni koneoppimisen menetelmä perustuu ratkaisemaan tämän ongelman erilaisilla lähentymistavoilla.[1]

Jäljittelyoppiminen perustuu esimerkin seuraamiseen. Tämä voidaan toteuttaa kineettisesti (engl. kinesthetic teaching), teleoperaatiolla tai havainnollisella opetuk-

sella (engl. observational learning). Kineettinen menetelmä perustuu robotin osien siirtämiseen haluttuihin sijainteihin ja tämän tallentamiseen tilannekuviksi (engl. snapshot). Näistä tilannekuvista muodostetaan rutiinille haluttavat liikeradat. Teleoperaatiolla robottia ohjataan manuaalisesti kauko-ohjainjärjestelmällä. Havainnollisen opetuksen menetelmä perustuu seuraamaan jotain opettajaa liikekaappausjärjestelmillä tai sensoreilla. Opettajan liikkeet yhdistetään robotin liikeratoihin. Jäljittelyoppimisen ongelmat muodostuvat, kun mikään opettaja ei kykene toteuttamaan tarvittavia liikkeitä. [1] [4]

Kontekstuaalinen rosvo -menetelmät perustuvat toimimaan järjestelmän dynamiikassa. Menetelmälle määritetään syötteet. Tätä kutsutaan järjestelmän kontekstiksi. Menetelmä päättää toimensa syötteiden johdosta. Järjestelmä ylläpitää arviota toimien tuottamasta palkkiosta. Kun järjestelmä valitsee toimen, järjestelmä nostaa tämän toimen valitsemisen todennäköisyyttä, jos tämä on hyödyllinen. Menetelmän ongelmaksi muodostuu tarve tasapainottaa "tutkiminen" ja "hyödyntäminen" (engl. exploration vs exploitation), eli uusien toimien yrittäminen vai luotettavien toimien tekeminen.[1] Kontekstuaalinen rosvo on Markovin päätösprosessin järjestelmä, joka hyödyntää vain yhtä tilaa,[7] käytännössä tämä on yksinkertainen versio vahvistusoppimisesta.

Vahvistusoppiminen on oppimismenetelmä, joka perustuu yritykseen ja erehdykseen. Järjestelmää palkitaan toimenpiteistä, jotka johtavat lähemmäs päämäärää ja rangaistaan tästä erkaantumisesta. Järjestelmällä on erillisiä syötteitä, joiden vaikutusta järjestelmä säätelee. Useasti tähän kuuluu dataa antureista, aiemmista päätöksistä ja muusta heuristiikasta. Robotiikassa tällainen menetelmä mahdollistaa kyvykkyyden, jota ei muilla menetelmillä saataisi teetettyä. Esimerkiksi on tilanteita, joissa tiettyä toimintoa ei voida mallintaa opettajan tai suoran ohjelmoinnin avulla. Hyötyä on myös silloin, kun pyritään löytämään optimaalinen ratkaisu ongelmaan, jota ei voida ratkaista analyyttisellä kaavalla. Oppimismallin avulla robotti

kykenee sopeutumaan ympäristön ja itsensä muutoksiin, kuten osien kulumiseen tai lämpenemiseen. [1]

2.2 Vahvistusoppiminen ja Markovin päätösprosessi

Klassiset lähestymistavat vahvistusoppimiseen perustuvat oletukseen ongelman esitettävyydestä Markovin päätösprosessina (engl. Markov decision process). Tämä tarkoittaa sitä, että jokainen mahdollinen tilanne voidaan esittää tilana s ja toimina a, jotka johtavat toiseen tilaan s'.[1] Robotti on esimerkki toimijasta (engl. agent), koska se on vuorovaikutuksessa maailman kanssa.[8] Robotti toimii abstraktisti "tila-avaruudessa" (engl. state-space), joka kattaa kaikki erilaiset tilat ja niistä toiseen tilaan muuntavat toimet (engl. action), joita robotilla voi toteuttaa. [9]

Robotiikassa monimutkaisia järjestelmiä kehitettäessä sen tutkimiseen tarvittava aika kasvaa eksponentiaalisesti tila-avaruuden koon mukaan. Tätä kutsutaan korkeaulotteisuudeksi (engl. high-dimensional), eli järjestelmän mahdolliset tilat kasvavat suhteessa toimijan mahdollisten toimien myötä. Käytännössä kaikkien mahdollisten tilojen läpikäynti tulee nopeasti ajallisesti mahdottomaksi. Tämän takia monet vahvistusoppimisen menetelmät perustuvat tilojen räätälöimiseen, eli rajoittamaan tila-avaruuden kokoa, vähentääkseen opettamiseen kuluvaa aikaa. Tässä yhteydessä vahvistusoppimisessa nousee keskeiseksi tila-avaruuden käsittelyn muuttaminen toimintatapa-avaruudeksi (engl. policy-space), sillä tämän korkeaulotteisuus on huomattavasti pienempi kuin tila-avaruuden. [4]

Oppimisprosessissa simuloidaan robotin jäljitelmä, jonka avulla arvioidaan erilaisia liikkeitä ja niiden toivottavuutta. Toimintatapa toimii suunnitelmana, miten siirtyä tiloista toisiin tiloihin.[10] Käytännössä toimintatapa on reittiopas, jota parannellaan vahvistusoppimisen prosessin kautta. Robotti valitsee toimintatavan pe-

rusteella toimen siirtyäkseen tila-avaruudessa tilasta toiseen. Miten toimijan toimia arvioidaan, riippuu menetelmästä, instanssin lopputuloksesta ja jokaisesta läpikäydystä tilasta. [4]

Käytännössä kaikki vahvistusoppiminen perustuu Markovin päätösprosessiin, jonka muodollinen kuvaus on $\{S, A, T, R, p(s_0), \gamma\}$. S tarkoittaa kaikkien tilojen s joukkoa, $s \in S$. A on toimijan mahdollisten toimien a joukko, $a \in A$. Ympäristö koostuu siirtymäfunktiosta $T : S \times A \to p(S)$ ja palkintofunktiosta, $R : S \times A \times S \to \mathbb{R}$.

Siirtymäfunktio on joukko todennäköisyyksiä kullekin toimelle johtaa toiseen tilaan. Palkintofunktio on joukko arvioita tilan siirtymästä toiseen. $p(s_0)$ tarkoittaa järjestelmän mahdollisten aloitustilojen s_0 satunnaista joukkoa. γ on alennusparametri, jolla voidaan muokata prosessia. [1] [10]

Siirtymäfunktion muodostama siirtymätodennäköisyys P määritetään (engl. transition probability)

$$T(s', a, s) = P(s'|s, a) \tag{2.1}$$

, joka esittää todennäköisyyttä siirtyä tilaan s' mikäli tilassa s tehdään toiminto a. Tämä prosessi pyrkii luomaan matemaattisen mallin siitä, miten jokin järjestelmä toimii, olettaen, että järjestelmä on deterministinen. Vaikka Markovin päätösprosessi olettaa piirteeksi järjestelmän olevan deterministinen, siirtymätodennäköisyydet mahdollistavat epävarmojen tilojen mallinnuksen. Todennäköisyysarvio supistaa epävarmat mahdollisuudet käsiteltäväksi painotteeksi.[1]

Toimintatapa on arvio siitä, mistä järjestelmän tulisi tehdä parhaan lopputuloksen saavuttamiseksi. Eri menetelmillä on erilaiset kriteerit muodostaa arvio. Toimintatapa esitetään symbolilla π. Deterministinen toimintatapa tekee saman toimen tilasta riippuen, tämä merkitään $a = \pi(s)$. Stokastisessa toimintatavassa toimi valitaan siirtymätodennäköisyyksien perusteella, mikä ilmaistaan

$$a \sim \pi(s, a) = P(a|s). \tag{2.2}$$

[1]

Toimitavan muokkaus perustuu palkintofunktioon (engl. reward function) R. Funktio ohjaa järjestelmää haluttuun suuntaan. Muokkaus toimii joko "palkintona", eli sitä säädellään todennäköisemmäksi valinnaksi, tai "rangaistuksena", jolloin sen valintaa vähennetään. Eri menetelmissä järjestelmän palkitseminen toimii eri tavoilla: nykyisen tilan perusteella $R = R(s)$, sen hetkisen tilan ja valitun toimen perusteella

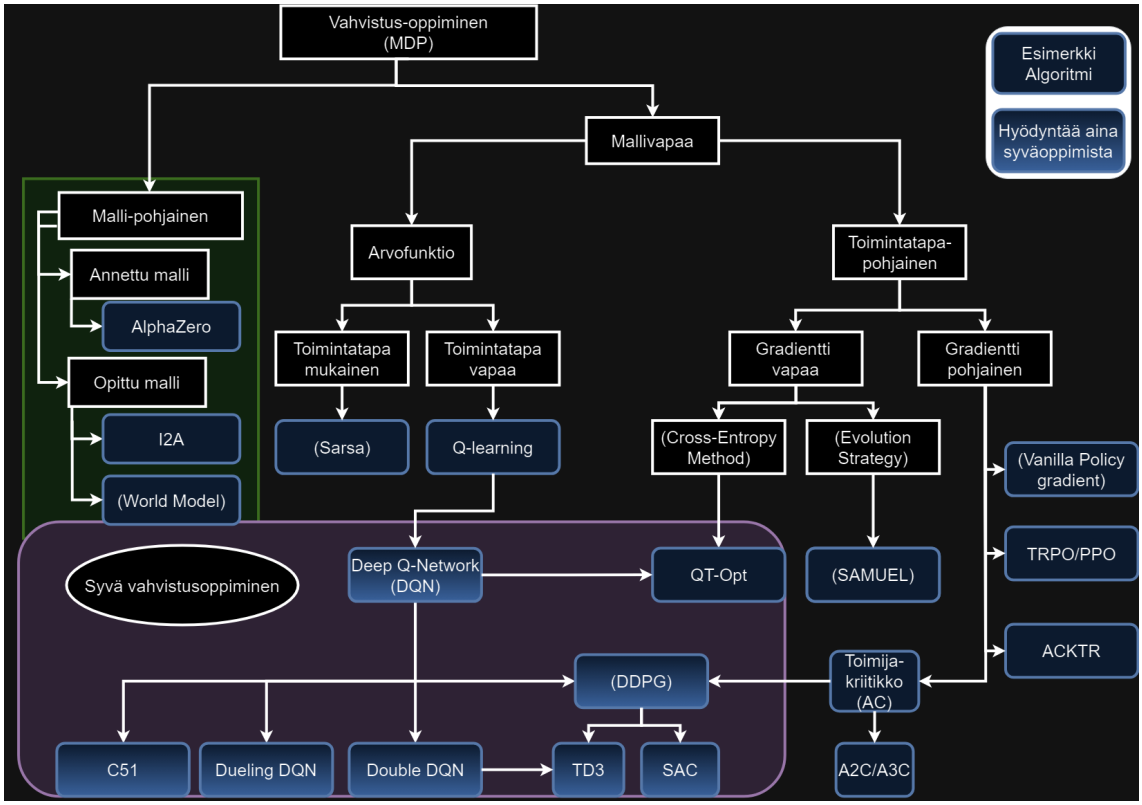
$$R = R(s, a) \tag{2.3}$$

tai ottaen huomioon siirtymän. [1]

$$R = R(s', a, s) \tag{2.4}$$

Vahvistusoppiminen jakautuu pääasiallisesti kahteen kategoriaan: mallivapaaseen (engl. model-free) ja mallipohjaiseen (engl. model-based).[1] Mallivapaan järjestelmän oppiminen perustuu jokaisen opittavan tilan läpi käymiseen. Tämän kaltaisella järjestelmällä ei ole mahdollisuuksia rajoittavaa arviota, joten ne muodostavat stokastisia toimitapoja.[11] Mallipohjainen järjestelmä sisältää mallin, jolla voidaan arvioida seuraavien tilojen mahdollisia lopputuloksia.[1] Malli on jokin muotoiltu ymmärrys toimijan tarpeista jossain ympäristössä. Mallipohjainen järjestelmä rajoittaa stokastisen prosessin mallin tulkinnoilla tila-avaruudesta. Mallipohjainen vahvistusoppiminen voi muodostaa stokastisen tai deterministisen toimintatavan.[11] Malli avustaa löytämään soveltuvan toimintatavan nopeammin, mutta se voi myös rajoittaa opetusprosessin mahdollisia lopputuloksia.[9] Stokastiset vahvistusoppimismenetelmät voivat perustua arvofunktioon (engl. value function) tai toimi-arvofunktioon (engl. q-value tai action-value function).[9] Toimi-arvofunktio

esitetään $Q(s, a)$, eli Q:n arvon määrittää tila ja toimi.[11]

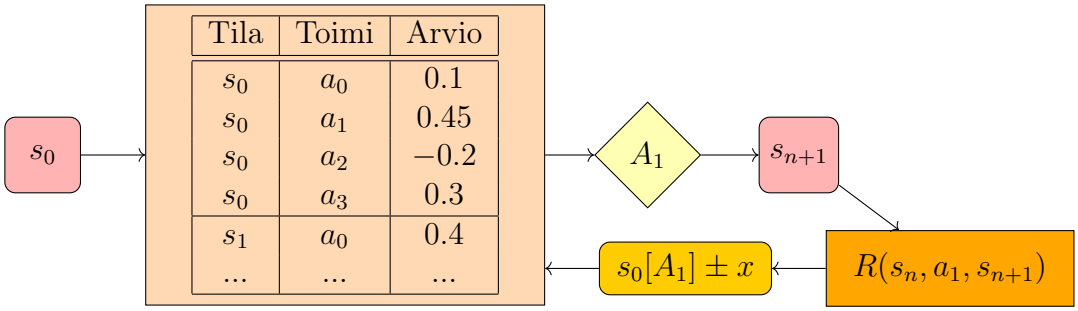


Kuva 2.1: Osa vahvistusoppimisen taksonomiastaKokoava lajittelu osasta vahvistusoppimismenetelmien taxonomiasta. Perustuu [12] taxonomiseen luokitteluun ja lisäyksiin hyödyntäen erillisiä lähteitä [13] [8] [14].

2.2.1 Mallivapaa vahvistusoppiminen

Mallivapaat menetelmät jakautuvat perinteisesti arvofunktiopohjaisiin (engl. value-based) ja toimintatapapohjaisiin (engl. policy-based).[1] Arvofunktiopohjaiset menetelmät keskittyvät muokkaamaan arvofunktiota, jolloin tiloja voidaan arvioida tarkemmin. Toimintatapapohjaiset menetelmät optimoivat toimintatapaa. Kolmas kategoria mallivapaata vahvistusoppimista on hybridi näiden kahden väliltä, toimijakriitikko menetelmät(engl. actor-critic), joissa hyödynnetään molempia lähentymistapoja. [11] [6]

Arvofunktiopohjaiset menetelmät jakautuvat kahteen eri kategoriaan: toiminta-



Kuva 2.2: Esimerkki yksinkertaisesta mallivapaasta vahvistusoppimisjärjestelmästä.

tavan mukaisiin (engl. on-policy) ja toimintatapavapaisiin (engl. off-policy) menetelmiin.[1] Näiden ero muodostuu siinä, seuraako järjestelmä toimintatapaa opetuksen aikana. Toimintatavasta vapaat menetelmät tutkivat tila-avaruutta ilman rajoituksia. Arvofunktiomenetelmien toimintatapa käytännössä seuraa arvofunktion maksimia. [1][6]

Yksi esimerkki arvofunktiopohjaisista menetelmistä on ajallisen erottelun (engl. temporal difference) menetelmät. Nämä tarkastelevat ajallisia virheitä (engl. Temporal error), eli jokaisen simulaatioaskeleen välisten arviointien eroja.[1] Tämä mahdollistaa järjestelmän arvofunktion muokkaamisen ennen simulaation loppumista. Muokkaaminen tapahtuu inkrementaalisesti askel askeleelta sen hetken arvioitavaksi tulevien tilojen perusteella. [8]

Toimintatapapohjaiset menetelmät jakautuvat gradienttipohjaisiksi (engl. gradient -based) ja gradienttivapaiksi (engl. gradient-free) menetelmiksi.[1] Gradienttipohjaiset järjestelmät keräävät otoksia (engl. sampling) dataa usealta opetusjaksolta. Tästä datasta muodostetaan todennäköisyysgradientti kaikista kerätyistä tilojen toimien arvoista. Tästä muodostuu esitys toimintatapa-avaruudesta, jonka sisällä voidaan laskea suotava muutoksen suunta.[1] Gradienttivapaat menetelmät hyödyntävät jotain muita heuristisia menetelmiä. Esimerkiksi tapa toteuttaa evolutionaalinen menetelmä on simuloida ympäristö monelle eri toimijalle ja kilpailuttamalla näiden tehokkuus.[11]

Yksi esimerkki gradienttipohjaisista menetelmistä on Monte Carlo -simulaatioon

perustuvat menetelmät. Nämä perustuvat tilojen mallintamiseen matemaattisesti Markovin päätösprosessin avulla. Otosten eri tilojen todennäköisyyksistä, ja niiden läheisyydestä haluttuun lopputulokseen, rakennetaan likimääräinen arvofunktio. Kun riittävä määrä esimerkkitilanteita on saavutettu, järjestelmä voi arvioida oletetun lopputuloksen mille tahansa tilalle.[11] Ideaalissa tilanteessa arvofunktio muotoutuu kuvaamaan ongelmaa täydellisesti, mutta usein funktio muodostaa likimääräisen arvion. [1]

Toimija-kriitikko-menetelmät voi ajatella pääasiallisesti gradienttipohjaisten menetelmien jatkeena.[11] Menetelmässä toimintatapa-avaruuden optimointiin yhdistetään arvofunktion oppiminen. Käytännössä tämä menetelmä on yhdistelmä arvofunktio- ja toimintatapamenetelmistä. Arvofunktion oppiminen mahdollistaa datan käytön, joka ei ole opittu toimintatavan kautta. [15] Arvofunktion parantaminen on käytännössä järjestelmän sovittamista ympäristöön ilman oletuksia, joita voi muodostua virheellisestä arvofunktiosta.

2.2.2 Mallipohjainen vahvistusoppiminen

Mallipohjainen vahvistusoppiminen laajentaa mallivapaata vahvistusoppimista ottamalla huomioon järjestelmän dynamiikat. Malli tarkoittaa Markovin päätösprosessin kuvaamaa mallia. Esimerkiksi tämä voi olla siirtodynamiikan (engl. transition dynamics) malli. [16] Hyödyntäen tämän kaltaista mallia voidaan laskea jokainen tarpeellinen liike saavuttaakseen jokin haluttu tila. Eli mallilla voidaan laskea mahdollinen tila-avaruuden hyöty usean askeleen päähän. [9][17] Miten malli rakennetaan, kutsutaan arkkitehtuuriksi. Mallinarkkitehtuuri voi tukea oppimista tai olla muuttumaton. [16]

Yksi kategoria malliarkkitehtuureja on suora malli (engl. forward model), joka ennustaa, mikä seuraava tila on, tämänhetkisen tilan ja toimen perusteella. $(s_t, a_t) \rightarrow s_{t+1}$ Oppiminen tällaisessa järjestelmässä on hyvin suoraviivaista, mutta useissa ti-

Tila	Toimi	Siirtymätodennäköisyys	Seuraava tila
s_0	a_0	$P(s_0\|s_0, a_0)$	s_0
s_0	a_1	$P(s_0\|s_1, a_1)$	s_1
...	...	...	...
s_0	a_n	$P(s_0\|s_n, a_n)$	s_n
s_1	a_0	$P(s_1\|s_0, a_0)$	s_0
s_1	a_1	$P(s_1\|s_1, a_1)$	s_1
...	...	...	...
s_n	a_n	$P(s_n\|s_n, a_n)$	s_n

Kuva 2.3: Esimerkki siirtymätodennäköisyyttä hyödyntävästä yhden askeleen mallipohjaisesta toimintatavasta. Käytännössä P on jokin todennäköisyys mikä sisältyy $[0, 1]$ väliseen joukkoon.

lanteissa malli ei tarjoa tarpeeksi tietoa seuraavan tilan arviointiin. Vastaava malliarkkitehtuuri, joka voi tarjota arvioita tulevasta tilasta, on usean askeleen ennustusmalli (engl. multi-step prediction model). Mallilla ennustetaan useita askelia eteenpäin pohjatilasta. Mutta sarja ennusteita on altis virhekertymälle, johtaen epäluotettavuuteen. Käänteismalli (engl. inverse model) tähtää arvioimaan, mikä toiminta tarvitaan siirtämään tila haluttuun tilaan. $(s_t, s_{t+1}) \rightarrow a_t$ Vaihtoehtoisesti käänteismalli voidaan rakentaa tunnistamaan, mikä toimi ja tila veivät tarkistettavaan tilaan. $s_{t+1} \rightarrow (s_t, a_t)$. [16] [10]

Oppivat arkkitehtuurit (engl. learning architecture) voivat muovata olemassa olevaa mallia, tai opettaa täysin uuden mallin. Käytännössä mallin opettaminen on ohjatun oppimisen ongelma. Oppivat arkkitehtuurit voidaan rakentaa yhdistämällä malli ja palautesäädin (engl. feedback controller). Palautesäädin on järjestelmästä tiedetyistä asioista rakennettu matemaattinen malli. Säädin käytännössä arvioi tilasta toiseen tilaan siirtymisen todennäköisyyttä. Tämän tarkoitus on tasata mallin virheitä ja ohjata sen koulutusta. Itse säätimen voi korvata mallivapaalla järjestelmällä. [16][10]

Eräs menetelmä toteuttaa oppiva arkkitehtuuri on suoramallinnus (engl. direct modeling). Tämä menetelmä perustuu sovittamaan malli järjestelmän alku- ja lop-

putuloksiin, eli se kartoittaa toimesta johdettuja tiloja. Suoramallinnus on yleisesti hyödyllinen rakentamaan suoria malleja. Sama arkkitehtuuri kykenee kouluttamaan usean askeleen ennustusmalleja lisätyllä heuristiikalla, joko neuroverkon tai todennäköisyysmenetelmien avulla. Epäsuora mallinnus (engl. indirect modeling) perustuu palautesäätimen oppimiseen. Arkkitehtuuri ei välitä, mitä toimia opetuksessa tehdään. Epäsuoran mallinnuksen järjestelmä seuraa pelkästään virheitä palauteohjaimen ja mallin arvioissa. Malli oppii järjestelmän dynamiikan, kun palauteohjaimen arvioissa ei ole enää virhettä. Tämän menetelmän täytyy tapahtua ympäristöstä vuorovaikutuksella kerätyllä datalla. [16]

Varsinkin simulaatiossa mallin kouluttaminen voi johtaa järjestelmiin, jotka eivät suoriudu todellisuudessa. [1] Tämä johtuu opetuksesta muodostuneista vääristymistä. Nämä voivat johtua alimallintamisesta (engl. under-modeling) tai simulaation puutteista. Kun malli ei täsmää ympäristöön, pienetkin virheet kertyvät vääristämään arviota askel askeleelta. [9][13]

2.2.3 Syvävahvistusoppiminen

Perinteisesti syväoppiminen (engl. deep learning) on erillinen kategoria koneoppimisessa vahvistusoppimisesta. Syväoppiminen perustuu monikerroksisten neuroverkkojen hyödyntämiseen. [11] Neuroverkot kykenevät esittämään korkeaulotteisen datan käytettävämmäksi matalaulotteisiksi dataksi.[2] Tätä hyödynnetään esimerkiksi tulkkaamaan kuvan, tekstin ja äänen hyödyllisiä ominaisuuksia. [10]

Neuroverkot ovat graafeja, joiden jokaisella noodilla on syötteilleen painoarvot (engl. weights) ja vinouma (engl. bias). Noodeja on kerroksittain (engl. layer). Noodien kytkennäisyys riippuu neuroverkon tyypistä. Täysin kytketyissä verkoissa jokainen aiemman kerroksen noodi on yhdistetty jokaiseen seuraavan kerroksen noodiin. Osittain kytketyissä verkoissa noodit ovat yhdistettyjä vain osaan seuraavan kerroksen noodeista. Noodin arvo lasketaan painoarvoilla muokattujen syötteiden ja vinou-

man summana. Tämän jälkeen arvo syötetään neuroverkolle määritettyyn aktivointifunktioon. Yksi yksinkertainen esimerkki aktivointifunktiosta on ReLU $f(x) = x$ jos $x > 0$ muulloin $f(x) = 0$. Neuroverkkojen kerroksiin kuuluu syötekerros, ulostulokerros ja näiden välillä olevat piilokerrokset. [18]

Neuroverkkoja on erilaisiin tarkoituksiin. Kolme yleisintä neuroverkkorakennetta, joita robotiikassa hyödynnetään, ovat eteenpäinkytketty neuroverkko (engl. feedforward neural network), rekursiivinen neuroverkko (engl. recurrent neural network) ja konvoluutioneuroverkot (engl. convolution neural network).[18]

Eteenpäinkytkettyverko on yksinkertaisin tapa toteuttaa syvä neuroverkko. Käytännössä neuroverkko koostuu useista tasoista noodeja, joista jokainen vaikuttaa vain seuraavan tason noodeihin. Rekursiivisen neuroverkon tasojen noodit voivat syöttää tuloksensa aikaisemmille tasoille, luoden eräänlaisen muistin. Takaisin syötetyt arvot kantautuvat seuraavan syötteen arviointiin. [18]

Konvoluutioverkot ovat yleensä eteenpäinkytkettyjä neuroverkkoja, joiden tarkoitus on hyödyntää suurta syötetasoa ja muuttaa se pieneksi hyödylliseksi ulostuloksi. Syöte voi koostua esimerkiksi kuvan jokaisesta pikselistä. Tämän tason jälkeen neuroverkkolle määritetään useita konvoluutiotasoja, joiden tarkoitus on tiivistää dataa konvoluutiolla. Näitä neuroverkkoja hyödynnetään varsinkin sensoridatan tulkintaan.[18]

Syvävahvistusoppiminen hyödyntää neuroverkkoja osana vahvistusoppimisjärjestelmää. Neuroverkot tarjoavat mahdollisuuden järjestelmälle hyödyntää korkeaulotteista raaka-dataa.[11] Tämä johtuu siitä, että neuroverkot oppivat koulutusdatan piirteitä. Tämä johtaa siihen, että koulutettu neuroverkko sisältää opetusdatan kaltaisen datan relaatiot, mikä on sovitettu haluttuihin ulostuloihin.

Esimerkiksi konvoluutioverkkoa on mahdollista hyödyntää tuottamaan järjestelmä, joka tulkitsee tilan ja tarpeellisen toimen hyödyntäen pelkkää videolähdettä.[17] Neuroverkkoja hyödynnetään paljon mallipohjaisessa vahvistusoppimisessa.[9] Mo-

nen ympäristön ja tehtävän kokonainen mallintaminen on hankalaa, tai käytännössä mahdotonta toteuttaa, koska nämä tilanteet useasti ovat yllättävän monimutkaisia.[1] Tämä on ongelma, johon neuroverkot ovat hyvin tehokkaita.

3 Simulaatio

3.1 Simulaation toteutus

Kaikelle koneoppimiselle datan kerääminen on hyvin tärkeää, varsinkin vahvistusoppimiselle. [1] Dataa voidaan kerätä monituisilla menetelmillä, mutta tehokkain tapa on käyttää simulaatiota. [19] Simulaatio on ympäristön jäljitelmä, joka on toteutettu matemaattisesti. Simulaattorin ajan kulku tapahtuu aika-askelina (engl. time-step), eli kyseisen alustan fysiikkamoottori laskee jokaisen liikkeen, joka simulaatiossa tapahtuu määritetyssä ajanjaksossa. [20][21] Usein simulaation tarkkuus on riippuvainen aika-askeleen koosta, mutta pieni aika-askel kasvattaa simulaatioon kuluvaa aikaa. Tämä johtuu siitä, että fysiikkamoottorin täytyy laskea useampia muutoksia maailmassa. [22]

Robotiikalle tärkeitä simuloitavia kohteita ovat jäykät ja pehmeät kappaleet (engl. rigid bodies & soft bodies), neste ja monimutkainen maasto, kuten rakeinen maa. Fysiikkamoottoreita, joita robotiikassa käytetään, kutsutaan dynamiikkamoottoreiksi. Tämän kaltaisen fysiikkamoottorin tarkoitus on mallintaa tarkasti dynamiikkaa, eli kappaleiden vuorovaikutuksen tuottamaa liikettä. Moneen tarkoitukseen robotti ja ympäristö toteutetaan jäykkinä kappaleina, eli nämä kappaleet eivät voi muovaantua. Edistyneemmät robotit useasti tarvitsevat monimutkaisempia ympäristöjä, kuten myös robotin osien ominaisuuksia. [23]

Jäykkien kappaleiden simuloinnissa keskeiset piirteet ovat kappaleen massa ja

kiihtyvyys. Kappaleiden muoto on merkityksellistä vain törmäystilanteissa. Koska monimutkaisen kappaleen törmäyksen tunnistusoperaatio on hyvin raskas, kappaleille määritetään törmäyspinta ja projektiopinta. Törmäyspinta on yksinkertaistettu ääriviivamalli kappaleesta, jonka avulla törmäystapahtumat lasketaan. Projektiopintaa hyödynnetään, jos simulaatio halutaan visualisoida, jolloin tarkemmalle kappaleen mallille voidaan piirtää tekstuuri. [23]

Robotit yleisesti mallinnetaan kokoelmana jäykkiä kappaleita, jotka on yhdistetty nivelillä, eli kiinnityskohdilla, joille on määritetty sallittuja liikeratoja. Kuluvista ja joustavista osista koostuville roboteille tarvitaan muovautuvia kappaleita, eli pehmeitä kappaleita. Käytännössä pehmeä kappale toteutetaan ohjelmoimalla jäykälle kappaleelle muovautuvuutta. Monien menetelmien erot lienevät painon keskipisteen laskemisessa. Vaikka maastot ovat käytännössä aina jäykkiä kappaleita, esimerkiksi lumen ja hiekan simulointiin hyödynnetään pehmeiden kappaleiden menetelmiä. Nesteen simulaatio toteutetaan yleisesti Navier-Stokesin yhtälöillä. [23]

Valitettavasti simulaatio ei kykene välttämättä vastaamaan todellisuutta, kaikkea käytöstä ei välttämättä kyetä mallintamaan simulaatioon. Tätä kutsutaan todellisuusrajaksi (engl. reality gap). Vahvistusoppimisjärjestelmä voi myös ylisovittua (engl. over fitting) liian muuttumattomassa simulaatiossa. Vastaus todellisuusrajaan ja ylisovitukseen on yleistää (engl. generalize) järjestelmää, eli laajentaa tilanteita, joissa se toimii. [24][25]

3.2 Simulaatioympäristöt

Simulaatioratkaisuna voidaan käyttää joko valmista alustaa tai tällaisen voi rakentaa kokoelmasta työkaluja. Tämän kaltainen kokoonpano vahvistusoppimisessa on simulaatioympäristö, eli kokonaisuus fysiikkamoottorista, ympäristöstä ja muista työkaluista. Ympäristö koostuu simulaatioon määritetystä maailmasta ja robotista. [23] Simulaation liitetään rajapinnan läpi "päättäjä" (engl. controller), joka koostuu

vahvistusoppimisjärjestelmästä. [1]

Robotiikkaan on luotu ohjelmointirajapintoja, joiden avulla määritellään yleisiä käytäntöjä roboteille. ROS (Robot Operating System) on työkalujen kokoelma ja rajapinta, jonka avulla rakennetaan robottijärjestelmiä. Tämän kokoelman tarkoituksena on tehdä robotit yhteensopiviksi ja uudelleenkäytettäviksi. URDF (Universal Robotic Description Format) ja SDF (Simulation Description Format) ovat yleisiä menetelmiä, joiden avulla robotti ja sen ympäristö kuvataan XML-tiedostoon kokonaisuudessaan. Yleisen kielen kautta voidaan robotteja simuloida helposti monessa eri simulaatioympäristössä tarpeen mukaan. [20]

Valitessa simulaatioympäristö on tärkeää arvioida sen ominaisuuksia. Esimerkiksi yksi kätevimmistä ominaisuuksista simulaattorille on toimia päättömänä (engl. headless) eli ilman käyttöliittymää, mikä vapauttaa resursseja pyörittämään useata opetusjaksoa samanaikaisesti. Tämä myös helpottaa simulaation automatisointia. Simulaattorin rajapinnan suunnittelu voidaan esimerkiksi tehdä ROS-rajapinnalla, jolloin se on suoraan yhteensopivia monen robottijärjestelmän kanssa. Jos simulaattorilla on täysin itsenäinen rajapinta, sen hyödyntäminen muodostuu helposti hankalaksi. [26][19]

Simulaatiokehykset (engl. simulation framework) toimivat välikappaleena simulaatioympäristön ja käyttäjän välillä. Kehys voi perustua tiettyyn simulaatioympäristöön, mutta moni kehys tukee useita dynamiikkamoottoreita ja ympäristöjä. Vahvistusoppimista ei aina suoraan tueta simulaatioympäristössä, joten data täytyy kerätä ja tulkita simulaattorin ulkopuolella. Kehys voi olla varsin hyödyllinen vahvistusoppimisprosessiin, koska oppimismallia voidaan suoraan muokata ja hallita tämän tasolla. [19]

4 Vahvistusoppiminen, simulaatio ja robotiikka

4.1 Haasteet

Robottien tarkka hallinta ja hyödyntäminen vaativat tehokkaita ohjausjärjestelmiä. Haastavuus muodostuu robottien monimutkaisuudesta. Jokainen robotti voi vaatia täysin omankaltaisen ratkaisun, johtuen erilaisista osista ja kokoonpanoista. Tämä tuottaa ohjaamisen vaikeudet; mitään täydellistä ohjausratkaisua ei voi luoda tietämättä järjestelmää täydellisesti. Vahvistusoppimisen avulla on mahdollista etsiä robotin dynamiikasta haluttua käytöstä. Prosessi vaatii paljon dataa, mikä täytyy kerätä vuorovaikutuksen avulla. Tämä onnistuu joko fyysisellä toteutuksella tai simulaatiolla.

Miten robottijärjestelmälle räätälöidään syötteet, jotka vaikuttavat toimintatapaan. Syötteet ovat keskeisiä sille, mitä vahvistusoppimismalli voi käytännössä oppia.[1] Mutta mitä enemmän robottilla on syötteitä, sitä kalliimpaa on jokainen askel simulaatiota. Tämä johtuu tarpeesta tuottaa dataa, joka vastaa todellisuutta jokaisessa uudessa tilassa. Lisäksi monet syötteet, kuten kamerasta tuleva video, ovat hankalia kvantifioida hyödylliseksi dataksi. Tähän ratkaisuksi muodostuu syväoppiminen.

Syvävahvistusoppiminen perustuu perinteisen arkkitehtuurin muokkaamiseen

neuroverkoilla. Koulutuksen myötä neuroverkot toimivat tiiviinä esityksinä robotin tai käsiteltävän tehtävän dynamiikasta. Sisäistetty dynamiikka arvioi, mitä kustakin tilasta ja toimesta seuraa, joten järjestelmä kykenee olettamaan tarvittavan toimen tuntemattomassa tilassa. Opetuksen datan laatu vaikuttaa opitun dynamiikan tarkkuuteen.

Käytännössä robotiikan tarpeet muodostuvat näiden hyvin laajoista liikeulottuvuuksista. Kaikki robotin ominaisuudet tuottavat tutkittavaa tila-avaruutta. "Tutki ja hyödynnä" (engl. exploration and exploitation) on yleinen koneoppimisen ongelma, joka kuvaa uusien ratkaisujen etsimistä ja nykyisten optimointia. Tutkiminen voi johtaa parempaan ratkaisuun tai tuhlata vain aikaa, jolla optimoidaan tämänhetkistä ratkaisua. Tämä johtaa myös tarpeeseen varmistaa vertauspiste robotin toiminnalle, eli suunnitella testejä. Tämä on varsinkin tärkeää, koska simulaatiosta kerätty data ei ole taatusti luotettavaa. [7] Tähän tärkeä työkalu on oppimisen yleistäminen.

4.2 Yleistäminen

Simulaatio ei valitettavasti ole täydellinen vastike todellisuudelle, mutta se on tehokas tapa tuottaa tarpeellista dataa opetukseen. Tekniikoita, joilla koulutettuja järjestelmiä sopeutetaan, kutsutaan yleistämiseksi. Yleistäminen on tarpeellista simulaatiota hyödyntävissä vahvistusoppimisjärjestelmissä, koska simulaatio ei vastaa tarkasti todellisuutta. Näiden menetelmien tarkoitus on painostaa järjestelmä oppimaan joustavampi ratkaisu ja varmistaa, ettei oppiminen ajaudu virheelliseksi.

Järjestelmän tunnistus (engl. system identification) toteutetaan tarkkailemalla todellisen robotin keräämää dataa ja hienosäätämällä simulaatiota vastaamaan kerättyä dataa. Vaikka simulaatiota hienosäädetään hyvin tarkasti, on todellisuuden tarkkuuden saavuttaminen hyvin hankalaa. Usein tämä johtaa puutteisiin, joita ei voitu huomioida, kuten mahdolliset olosuhteet, joita ei simuloitu. Robottiin voivat

vaikuttaa kosteus, lämpötila ja osien kuluminen. Myös sensorien näkymien tarkka simulointi on haastavaa. [5]

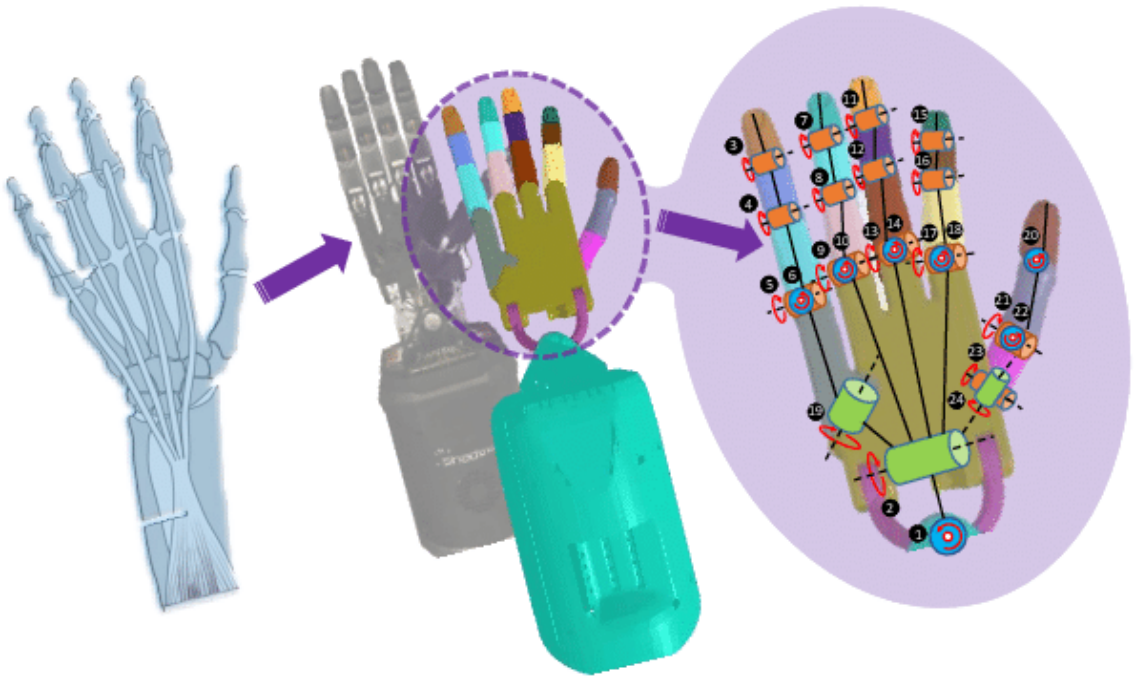
Toimintapiirin satunnaistaminen (engl. domain randomization) on hyödyllinen tapa yleistää vahvistusoppimisjärjestelmää. Toimintapiirillä tarkoitetaan järjestelmän tuntemaa ympäristöä, eli joukkoa ongelmia, joita järjestelmä on oppinut ratkaisemaan. Menetelmässä käytetään satunnaisia alkuarvoja simulaatiossa. Tämä auttaa toimijaa todennäköisemmin oppimaan rutiinin tärkeät piirteet. Nämä voivat olla visuaalisia tai dynaamisia. Esimerkiksi visuaalisia satunnaistettuja muuttujia voivat olla kameran sijainti, valojen määrä, käsiteltävien esineiden sijainti, muoto ja tekstuuri. Syvävahvistusoppimisella on mahdollista rakentaa neuroverkkoja, joilla voidaan tunnistaa esineitä konenäöllä, mikä helpottaa prosessia. Dynaamisia muuttujia voisivat olla esineiden koko, paino ja pintojen kitka. [24][5]

Satunnaistamista voidaan soveltaa muuhunkin kuin aloitusympäristön muuttamiseen, esimerkiksi tuottamalla satunnaisia impulsseja simuloituihin esineisiin. Vaihtoehtoisesti toimijan palkkio voidaan tehdä meluisaksi, palkkio vastaa enemmän todellisuutta olemalla hieman epätarkka. [5] [25]

Muita menetelmiä, joita voidaan hyödyntää, ovat toimialan sovitus (engl. Domain adaption) ja metaoppiminen (engl. Meta-learning). Molemmat menetelmät hyödyntävät dataa järjestelmän ulkopuolelta. Meta-oppimisessa hyödynnetään käytettävän datan luokittelua ja lisätietoja. Toimintapiirin sovituksessa hyödynnetään esimerkiksi valmiiksi koulutettua mallia järjestelmän hienosäätämiseksi. [24][5]

4.3 Käsittelyrobotiikka

Yleisiä robotiikan ongelmia, joihin vahvistusoppiminen toimii tehokkaana ratkaisuna, ovat esineiden käsittely (engl. manipulation), suunnistus (engl. navigation) ja monitoimijajärjestelmät. (engl. multi-agent systems) [7] Esimerkiksi esineiden käsittelyssä vahvistusoppiminen mahdollistaa monimutkaisten ja mukautuvien käsitte-



Kuva 4.1: "Shadow Dexterous Hand-robottikäden liikeradat[28]

lytaitojen kehittämisen. Nämä taidot ovat tärkeitä teollisuusautomaatiossa. Mutta pelkkä käsittelytaito ei useasti riitä hyödylliselle robotille. Tämä taito täytyy useasti integroida toimimaan liikkeen ja muiden robottien kanssa, muodostamaan monimutkaisia järjestelmiä.

Esineiden käsittely on ihmisille varsin ominainen taito, mutta robotille tämä voi olla varsin hankalaa. Yksi esimerkki tästä on robottikäsi, jonka tarkoitus on käännellä kuvioitua kuutiota halutuille sivuille [25], tai käsitellä useita erilaisia esineitä [27]. Käsirakenteet voivat olla yksinkertaisia tai monimutkaisia. Yksinkertainen "käsi" on käytännössä puristin ja sijoitusraaja. Yksinkertaisille robottiraajoille muodostuu usein vain 5 vapausastetta: "puristin", "ranne", "kyynärpää", "olkapää" ja horisontaaliliikerata. Monimutkaisen käsirakenteen yksi sormi saattaa sisältää yhtä monta vapausastetta kuin yksinkertainen robottikäsi. Tämä johtaa järjestelmään, jota on käytännössä mahdotonta ohjelmoida hyödyntämään jokaista liikerataa tehokkaasti.

4.3.1 Monimutkainen esineiden käsittely

Yksi esimerkkitutkielma monimutkaisesta robottikäsijärjestelmästä on Open AI:n teettämä "Learning dexterous in-hand manipulation"[25]. Tutkielman järjestelmä perustuu täysin simulaatiossa tapahtuvaan opetukseen, joka siirretään fyysiseen viisisormiseen robottiin. Fyysinen järjestelmä koostuu kolmesta kamerasta, 16 liikkeenkaappauskamerasta ja kuvassa 4.1 esitetystä robotista. Liikkeenkaappauskamerat toimivat kalibrointijärjestelmänä robotin sormien sijainnille. Tämä johtuu sormien sisäisten sensorien tilariippuvaisesta melusta, jota on hankala simuloida. Järjestelmän tavoite on kääntää kuutiota halutulle sivulle tehokkaasti kämmenen sisällä.[25]

Järjestelmän arkkitehtuuri perustuu yhdistelmään konvoluutioneuroverkkoja ja eräänlaiseen rekursiiviseen neuroverkkoon. Konvoluutioverkon tarkoitus on oppia tunnistamaan kuution ja käden sijainti toteuttaen konenäköä. Järjestelmä kouluttaa kahta eri rekursioverkkoa, toimintatapaa ja arvioijaa. Arvioija saa syötteikseen puhtaan kuution ja robotin sijainnit. Toimintatapaa koulutetaan vain "meluisilla" syötteillä. [25]

Koska todellisuudessa robottiin vaikuttavat erilaiset ympäristötekijät, joita on hankala tai mahdotonta simuloida, tärkeäksi osaksi opetusta muodostuu yleistäminen. Kyseisessä toteutuksessa yleistämistä toteutetaan toimintapiirin satunnaistamisella ja simulaattorin kalibroinnilla. Simulaation kalibrointi onnistuu vertailemalla oikean robotin ja simuloidun robotin liikkeitä valmiilla toimintatavalla. Parametrejä hienosäädetään niin, että simuloitu robotti ja todellisen robotin liikkeet vastaavat toisiaan. Näin jokainen simuloidun robotin liikkuva osa on täsmää tarkemmin kulumaa ja erilaisia marginaalisia tuotantovirheitä, joita todellisilla roboteilla tulee todennäköisesti olemaan. [25]

Simulaation melu tuotetaan kolmella eri tavalla, ja melu keskittyy hienosäädettyjen arvojen ympärille. Tämä datamelu tuotetaan pieninä millimetrien virheinä konenäössä, käden paikkatiedossa ja lievinä muutoksina simulaatioympäristön fysii-

kassa. Fysiikkamuutokset toteutetaan käsiteltävän esineen koon muutoksena, painovoiman nostamisena tai laskemisena, erilaisten kitkojen lievinä säätöinä ja robotin liikeratojen lievänä muutoksena. Tämän lisäksi robottiin kohdistetaan pientä kiihdytystä satunnaisesti. Jokainen esine satunnaistetaan simulaation alussa. Tämä koostuu tekstuurien ja värien, robotin sijainnin ja asennon, kuten myös kameroiden ja valojen sijaintien muutoksesta. [25]

Tämä järjestelmä toteuttaa tehokkaan toimintatavan, joka kykenee hyödyntämään tehokkaasti käden toimintoja. Varsinkin järjestelmän kyky hyödyntää pelkkää simulaatio-opetusta tekee opetuksesta tehokkaampaa ja helpompaa. Tämänkaltainen käsittely ei varsinaisesti ole itsessään tarpeeksi muodostaakseen hyödyllistä järjestelmää. Yksi haluttavimmista toiminnoista roboteille, esineiden käsittelyn lisäksi, on liikkuminen.

4.3.2 Liikkuvat ja yhteistyötä tekevät robotit

Liikkuva robotiikka (engl. mobile robotics) hyödyntää vahvistusoppimista pääasiallisesti kahteen tarpeeseen: liikkumiskyky (engl. locomotion) ja navigaatio. Liikkumiskyvyn tuottaminen yksinkertaisille roboteille on useasti kovakoodattua, mutta vahvistusoppimisen tarve ilmenee nopeasti monimutkaisille järjestelmille. Itsessään navigaatio robotiikassa ei ole vain reittien etsimistä, pikemminkin suunnittelua ja reagointia dynaamisessa ympäristössä.[29] Tämä tarkoittaa sitä, että reitin löytämisen lisäksi robotin täytyy arvioida reittiin vaikuttavia muuttujia.

Liikkuvat robotit eivät ole tehty vain liikkumaan. Liikkuminen on useasti vain askel suuremmasta rutiinista, kuten esineiden siirtämisessä. Varsinaisesti tämän kaltaisten järjestelmien tarkoitus on saavuttaa täysin itsenäisiä järjestelmiä.[29] Tähän keskeiseksi nousee tuottaa robotille tehokas ja joustava kyky liikkua ympäristössään. Sisätilat saattavat pääasiallisesti pysyä samankaltaisina, mutta robotin täytyy kyetä reagoimaan yllättäviin esteisiin.[29] Ulkona toimivat robotit tarvitsevat kykyä

toimia erilaisissa maastoissa ja sääolosuhteissa. Raajallisen robotin täytyy kyetä esimerkiksi sopeuttamaan askeltaan kumpuisessa maastossa.[30]

Yksittäiset robotit useasti hyödyntävät "hierarkkisia tehtäväarakenteita".[1] Esimerkiksi robotille koulutetaan yksi toimintatapa, jonka tehtävänä on robotin siirtäminen haluttuun sijaintiin, niin sanottu "korkeamman tason ohjain" (engl. high-level controller). Ja toinen toimintatapa, niin sanottu "alemman tason ohjain" (engl. low-level controller), jota ylemmän tason ohjain delegoi tarpeen mukaan käsittelemään esineitä. Molempien ohjainten koulutuksessa täytyy tiedostaa toisen vaikutus.

Monitoimijavahvistusoppiminen (engl. multi-agent reinforcement learning)[11], tai monirobottijärjestelmät (engl. multi-robot systems)[29], perustuvat usean robotin yhdessä muodostamiin kokoonpanoihin. Useat toimijat ympäristössä tekevät siitä dynaamisen, joten robotin täytyy kyetä reagoimaan toisiin robotteihin. Järjestelmän robottien tarkoitus on toimia yhdessä, joten niiden täytyy kyetä toteuttamaan rutiininsa tehokkaasti ja ilman häiriötä toisilleen.[11]

Monirobottijärjestelmiä voi kuvailla parvina. Näiden määrittävät piirteet ovat hierarkia, rakenne ja vuorovaikutus. Perinteisimpiä parvia ovat järjestelmät, jotka koostuvat samankaltaisista roboteista, kuten drooneista, nämä ovat homogeenisiä parvia. Tämän kaltaiset parvet ovat itsestään hajautettuja, jokainen toimija koordinoi toisten läheisten robottien kanssa. [14] Toisen kaltainen parven tyyppi on heterogeeninen parvi. Tämänkaltaiset parvet koostuvat usein erilaisista roboteista. Nämä järjestelmät voivat olla hierarkkisia tai hajautettuja. Esimerkiksi tehdasrobotit toimivat yhteistyössä keskitetyllä logiikalla. [28][14]

5 Yhteenveto

Vahvistusoppiminen on keskeinen osa modernia robotiikkaa. Se mahdollistaa aiempaa tehokkaamman ja laajemman robottien hyödyntämisen. Tämän tutkielman tarkoitus oli tiivistää tarpeellista pohjatietoa alueittain vahvistusoppimisen hyödyntämisestä robotiikassa. Tutkielma pyrki vastaamaan tähän seuraavilla tutkimuskysymyksillä: TK1. *Mitä on vahvistusoppiminen?*, TK2. *Miten simulaatiota hyödynnetään tässä prosessissa?* ja TK3. *Mitä hyötyä vahvistusoppimisesta on robotiikassa?*.

TK1 Vahvistusoppiminen on laaja aihealue koneoppimisen alla. Käytännössä vahvistusoppiminen on menetelmä etsiä täydellistä hallintaskeemaa toimintatavan muodossa. Vahvistusoppiminen perustuu Markovin prosessiin, joka on matemaattinen mallinnustapa tarkastella toimijan ja ympäristön vuorovaikutusta. Vahvistusoppiminen tarkastelee toimijan kykyä saavuttaa haluttuja tiloja toimintatavan valitsemien toimien kautta. Tilan hyödyllisyydestä muodostetaan arvio, jota hyödynnetään säätelemään toimintatapaa. Muutos vähentää tai kasvattaa toiminnan valinnan todennäköisyyttä. Tämä on useasti kuvattu "yrityksen ja erehdyksen"menetelmänä.

Yksi vahvimmista syistä, miksi vahvistusoppimista hyödynnetään robotiikassa on se, että robotit ovat esitettäviä Markovin prosessin avulla. Vahvistusoppiminen jakautuu pääasiallisesti kahteen kategoriaan: mallipohjaiseen ja mallivapaaseen. Mallivapaat järjestelmät parantavat toimintatapaa tarkastelemalla saavutettujen tilojen arvoa. Mallipohjaiset järjestelmät hyödyntävät malleja ympäristöstä arvioimaan tutkittavien tilojen arvoa. Tämä rajoittaa tutkittuja tiloja, mutta mahdollistaa no-

peampaa ja tarkempaa oppimista. Huonosti täsmäävä malli ympäristön vuorovaikutuksesta voi hidastaa ja heikentää oppimisen tulosta. Tämän takia useat mallipohjaiset menetelmät eivät säätele itse toimintatapaa, mutta säätelevät tilojen arvion tarkkuutta.

TK2 Kaikki koneoppiminen perustuu datan hyödyntämiseen. Koska vahvistusoppiminen perustuu vuorovaikutuksen mallintamiseen, hyödyllisen koulutusdatan tuottaminen vaatii ympäristön ja toimijan. Robotiikassa suora vahvistusoppiminen on ongelmallista, mikä johtuu mahdollisuudesta robotin vaurioitua tai vahingoittaa ihmistä oppimisprosessin aikana. Tämän takia robotiikassa hyödynnetään simulaatiota oppimisprosessissa.

Simulaatio on tehokas tapa tuottaa dataa vahvistusoppimiseen. Tuottamalla säädeltävä ympäristö ja toimija, simulaatio mahdollistaa tehokkaan kouluttamisen erilaisissa olosuhteissa ja kokoonpanoissa. Simulaatiolla on mahdollista kouluttaa, laskentatehon rajoissa, erilaisilla olosuhteilla samanaikaisesti useilla simuloiduilla roboteilla. Simulaatio ei kuitenkaan tuota täydellistä dataa, johtuen pienistä virheistä ja mallintamattomista luonnonilmiöistä todellisuudessa. Kosteus, lämpötila ja muut ympäristötekijät, kuten myös osien kuluminen, voivat olla hankalia tuottaa tarpeeksi suurella variaatiolla vastaamaan todellisten käyttöolosuhteiden moninaisuutta. Simulaation hyödyntäminen on kuitenkin hyödyllistä säästämään mitattomasti aikaa fyysisesti kerättyyn dataan verrattuna.

TK3 Vahvistusoppimista hyödynnetään robotiikassa opettamaan käsittelytaitoja ja liikuntadynamiikkaa. Miten robotin rutiini käsittelee esineitä tai liikkuu ympäristössä, on merkityksellistä. Ohjelmoitu rutiini ei välttämättä kata jokaista tilannetta, ja ohjelmoija ei kykene tietämään jokaisen ympäristötekijän vaikutuksia robottiin. Vahvistusoppimisen hyöty on tuottaa joustavia rutiineja. Tämä muodostuu vahvistusoppimisella luodulla toimintatavalla. Opetus muodostaa tehtävään soveltuvan

toimintatavan, joka arvioi sopivan liikkeen jokaiseen osaan rutiinia.

Syvävahvistusoppiminen hyödyntää neuroverkkoja vahvistusoppimisjärjestelmän osana. Robotiikassa neuroverkot mahdollistavat monimutkaisten anturien hyödyntämisen. Tämän kaltaisten anturien tulkkaaminen on perinteisesti laskennallisesti kallista. Esimerkiksi kameroiden dataa on tarpeellista tulkita esineiden tai ympäristön tunnistukseksi. Konenäön yhdistäminen vahvistusoppimiseen mahdollistaa monia monimutkaisia käsittelytehtäviä erilaisissa tilanteissa. Neuroverkot itsessään ovat vain tapa tallentaa dataa, mutta vahvistusoppimisen kautta neuroverkosta muovataan toimintatapa tai robotin ja ympäristön dynamiikan esitys.

Lopuksi Vahvistusoppimisen ja simulaation käyttötarkoitukset robotiikassa ovat laajoja. Tämä kirjallisuuskatsaus oli vain pintapuolinen tarkastelu aiheesta. Mahdollisia lisätutkimuksen alueita muodostuu jokaiselle tutkimuskysymykselle. Vahvistusoppimisen erilaisten menetelmien todellisen taksonomian tarkastelu voisi mahdollistaa aihealueen semantiikan selventämisen. Simulaatio-oppimisen yleistämisen menetelmien tarkastelu laajassa kontekstissa on mahdollisesti hyödyllinen tutkimuksen kohde. Lisäksi ei-hierarkisten robottien hallintajärjestelmien toteutus voisi olla mielenkiintoinen tutkimuksen kohde.

Lähdeluettelo

[1] J. Kober, J. A. Bagnell ja J. Peters, ”Reinforcement learning in robotics: A survey”, *The International Journal of Robotics Research*, vol. 32, nro 11, s. 1238–1274, syyskuu 2013, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364913495721. url: http://journals.sagepub.com/doi/10.1177/0278364913495721 (viitattu 13. 02. 2023).

[2] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin ja B. Dresp-Langley, ”Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review”, *Robotics*, vol. 10, nro 1, s. 22, 24. tammikuuta 2021, ISSN: 2218-6581. DOI: 10.3390/robotics10010022. url: https://www.mdpi.com/2218-6581/10/1/22 (viitattu 27. 03. 2023).

[3] N. Sünderhauf, O. Brock, W. Scheirer et al., *The Limits and Potentials of Deep Learning for Robotics*, 18. huhtikuuta 2018. arXiv: 1804.06557[cs]. url: http://arxiv.org/abs/1804.06557 (viitattu 20. 03. 2024).

[4] P. Kormushev, S. Calinon ja D. Caldwell, ”Reinforcement learning in robotics: Applications and real-world challenges”, *Robotics*, vol. 2, nro 3, s. 122–148, 5. heinäkuuta 2013, ISSN: 2218-6581. DOI: 10.3390/robotics2030122. url: http://www.mdpi.com/2218-6581/2/3/122 (viitattu 13. 02. 2023).

[5] W. Zhao, J. P. Queralta ja T. Westerlund, ”Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey”, teoksessa *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Canberra, ACT, Australia:

IEEE, 1. joulukuuta 2020, s. 737–744, ISBN: 978-1-72812-547-3. DOI: 10.1109/SSCI47803.2020.9308468. url: https://ieeexplore.ieee.org/document/9308468/ (viitattu 13. 02. 2023).

[6] J. Ramírez, W. Yu ja A. Perrusquía, "Model-free reinforcement learning from expert demonstrations: A survey", *Artificial Intelligence Review*, vol. 55, nro 4, s. 3213–3241, huhtikuu 2022, ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-021-10085-1. url: https://link.springer.com/10.1007/s10462-021-10085-1 (viitattu 07. 01. 2024).

[7] T. Zhang ja H. Mo, "Reinforcement learning for robot research: A comprehensive review and open issues", *International Journal of Advanced Robotic Systems*, vol. 18, nro 3, s. 172 988 142 110 073, 1. toukokuuta 2021, ISSN: 1729-8814, 1729-8814. DOI: 10.1177/17298814211007305. url: http://journals.sagepub.com/doi/10.1177/17298814211007305 (viitattu 02. 04. 2024).

[8] E. F. Morales, R. Murrieta-Cid, I. Becerra ja M. A. Esquivel-Basaldua, "A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning", *Intelligent Service Robotics*, vol. 14, nro 5, s. 773–805, marraskuu 2021, ISSN: 1861-2776, 1861-2784. DOI: 10.1007/s11370-021-00398-z. url: https://link.springer.com/10.1007/s11370-021-00398-z (viitattu 27. 03. 2023).

[9] A. S. Polydoros ja L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics", *Journal of Intelligent & Robotic Systems*, vol. 86, nro 2, s. 153–173, toukokuu 2017, ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-017-0468-y. url: http://link.springer.com/10.1007/s10846-017-0468-y (viitattu 19. 10. 2023).

[10] T. M. Moerland, J. Broekens, A. Plaat ja C. M. Jonker, *Model-based Reinforcement Learning: A Survey*, 2022. arXiv: 2006.16712 [cs.LG].

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage ja A. A. Bharath, ”A Brief Survey of Deep Reinforcement Learning”, *IEEE Signal Processing Magazine*, vol. 34, nro 6, s. 26–38, marraskuu 2017, ISSN: 1053-5888. DOI: 10.1109/MSP.2017.2743240. arXiv: 1708.05866[cs,stat]. url: http://arxiv.org/abs/1708.05866 (viitattu 01. 04. 2023).

[12] H. Zhang ja T. Yu, ”Taxonomy of reinforcement learning algorithms”, teoksessa *Deep Reinforcement Learning*, H. Dong, Z. Ding ja S. Zhang, toim., Singapore: Springer Singapore, 2020, s. 125–133, ISBN: 9789811540943 9789811540950. DOI: 10.1007/978-981-15-4095-0_3. url: http://link.springer.com/10.1007/978-981-15-4095-0_3 (viitattu 20. 03. 2024).

[13] J. Achiam, ”OpenAI Spinning Up”, *2018, OpenAI.*, yhteistyössä P. Abbeel, Revision 038665d6 2018. url: https://spinningup.openai.com/en/latest/index.html.

[14] M.-A. Blais ja M. A. Akhloufi, ”Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators”, *Cognitive Robotics*, vol. 3, s. 226–256, 2023, ISSN: 2667-2413. DOI: https://doi.org/10.1016/j.cogr.2023.07.004. url: https://www.sciencedirect.com/science/article/pii/S2667241323000241.

[15] A. T. Azar, A. Koubaa, N. Ali Mohamed et al., ”Drone deep reinforcement learning: A review”, *Electronics*, vol. 10, nro 9, s. 999, 22. huhtikuuta 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10090999. url: https://www.mdpi.com/2079-9292/10/9/999 (viitattu 20. 03. 2024).

[16] D. Nguyen-Tuong ja J. Peters, ”Model learning for robot control: A survey”, *Cognitive Processing*, vol. 12, nro 4, s. 319–340, marraskuu 2011, ISSN: 1612-4782, 1612-4790. DOI: 10.1007/s10339-011-0404-1. url: http://link.springer.com/10.1007/s10339-011-0404-1 (viitattu 27. 03. 2023).

[17] D. Ha ja J. Schmidhuber, "World Models", 28. maaliskuuta 2018. DOI: 10.5281/zenodo.1207631. arXiv: 1803.10122[cs,stat]. url: http://arxiv.org/abs/1803.10122 (viitattu 20. 11. 2023).

[18] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural Networks*, vol. 61, s. 85–117, tammikuu 2015, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. url: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[19] F. P. Audonnet, A. Hamilton ja G. Aragon-Camarasa, *A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2*, 13. huhtikuuta 2022. arXiv: 2204.06433[cs]. url: http://arxiv.org/abs/2204.06433 (viitattu 19. 10. 2023).

[20] M. Santos Pessoa De Melo, J. Gomes Da Silva Neto, P. Jorge Lima Da Silva, J. M. X. Natario Teixeira ja V. Teichrieb, "Analysis and Comparison of Robotics 3D Simulators", teoksessa *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, Rio de Janeiro, Brazil: IEEE, lokakuu 2019, s. 242–251, ISBN: 978-1-72815-434-3. DOI: 10.1109/SVR.2019.00049. url: https://ieeexplore.ieee.org/document/8921035/ (viitattu 19. 10. 2023).

[21] T. Erez, Y. Tassa ja E. Todorov, "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX", teoksessa *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA: IEEE, toukokuu 2015, s. 4397–4404, ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139807. url: http://ieeexplore.ieee.org/document/7139807/ (viitattu 19. 10. 2023).

[22] M. Körber, J. Lange, S. Rediske, S. Steinmann ja R. Glück, *Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning*, 2021. arXiv: 2103.04616 [cs.RO].

[23] C. K. Liu ja D. Negrut, "The role of physics-based simulators in robotics", *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, nro 1, s. 35–58, 3. toukokuuta 2021, ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-072220-093055. url: https://www.annualreviews.org/doi/10.1146/annurev-control-072220-093055 (viitattu 20. 04. 2024).

[24] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba ja P. Abbeel, *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*, 20. maaliskuuta 2017. arXiv: 1703.06907[cs]. url: http://arxiv.org/abs/1703.06907 (viitattu 30. 03. 2023).

[25] O. M. Andrychowicz, B. Baker, M. Chociej et al., "Learning dexterous in-hand manipulation", *The International Journal of Robotics Research*, vol. 39, nro 1, s. 3–20, tammikuu 2020, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364919887447. url: http://journals.sagepub.com/doi/10.1177/0278364919887447 (viitattu 13. 02. 2023).

[26] A. Afzal, D. S. Katz, C. L. Goues ja C. S. Timperley, *A Study on the Challenges of Using Robotics Simulators for Testing*, 2020. arXiv: 2004.07368 [cs.RO].

[27] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz ja D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *The International Journal of Robotics Research*, vol. 37, nro 4, s. 421–436, huhtikuu 2018, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364917710318. url: http://journals.sagepub.com/doi/10.1177/0278364917710318 (viitattu 01. 04. 2023).

[28] Y. Chen, T. Wu, S. Wang et al., *Towards Human-Level Bimanual Dexterous Manipulation with Reinforcement Learning*, kesäkuu 2022. DOI: 10.48550/arXiv.2206.08686.

[29] L. C. Garaffa, M. Basso, A. A. Konzen ja E. P. De Freitas, ”Reinforcement Learning for Mobile Robotics Exploration: A Survey”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, nro 8, s. 3796–3810, elokuu 2023, ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2021.3124466. url: https://ieeexplore.ieee.org/document/9612713/ (viitattu 02. 04. 2024).

[30] Á. Belmonte-Baeza, J. Lee, G. Valsecchi ja M. Hutter, ”Meta Reinforcement Learning for Optimal Design of Legged Robots”, *IEEE Robotics and Automation Letters*, vol. 7, nro 4, s. 12 134–12 141, lokakuu 2022, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3211785. arXiv: 2210.02750[cs,eess]. url: http://arxiv.org/abs/2210.02750 (viitattu 19. 10. 2023).